Return Curvature Memory
AVC - 24 January 2026

A Minimal Translation Table for Existing Formalisms

Physical formalisms routinely compare a system to itself after traversal, but rarely consider this comparison to be a primitive operation.

Nothing here redefines existing concepts; it only makes explicit a comparison operation already performed within them.

We refer to this operation as return, by the standard grammatical definition of the word.

In many domains, return is implicit rather than named. It appears through phase closure, parallel transport, holonomy, curvature, or identity along a path. Although these concepts differ in representation, they all perform the same basic operation: a system is transported, and the result is compared to its initial state. Whether this comparison is exact, and whether any discrepancy accumulates, determines the structure of the formalism.

This paper isolates return as a primitive operation and presents a minimal classification based on two questions:
(1) Is return exact or non-exact?
(2) If non-exact, does the remainder accumulate?

The classification yields a simple 2×2 table.

	Exact return	Non-exact return
No accumulation	identity (trivial/flat)	phase remainder
Accumulation	constrained closure	accumulated memory (curvature)

This table is not an interpretation and introduces no new structure. It functions only as a translation device for existing work.

The following examples are cited only to show where return already operates implicitly, not to suggest reinterpretation or priority.

In quantum mechanics, return commonly appears as a unitary operator acting on a state after traversal in parameter or configuration space. Exact return corresponds to identity (or identity up to a global phase), while non-exact return produces a phase or rotation. When such remainders

do not accumulate, return remains local. When they do accumulate under repeated traversal, the effect is encoded algebraically through non-commuting operators.

In general relativity, return is encoded geometrically. Parallel transport of vectors or frames around closed loops compares a system to itself after traversal. Non-exact return appears as curvature: a failure of exact closure that accumulates and constrains future traversal. When accumulation is fixed, memory is encoded as geometry rather than as an operator acting on a state.

Light-like propagation occupies a limiting case. Along null trajectories, return is exact in the sense that no proper-time remainder accumulates. Traversal occurs without memory accumulation, placing such paths in the “exact return / no accumulation” regime of the table.

The table does not assert equivalence between formalisms, nor does it propose a unification. It does not modify existing equations or introduce new observables. Its only purpose is grammatical: to make explicit a shared operation already present across domains.

By naming return and classifying how its remainder behaves, the table clarifies why matrix representations, non-commutation, and curvature appear ubiquitously wherever systems must be compared to themselves after traversal. Concepts such as measurement, interpretation, choice, or dynamics are not addressed here and must enter elsewhere.

This work offers no explanation. It offers only a translation table.

Appendix A: Familiar Equations in Return Grammar

This appendix lists several standard equations and notes how they already encode return as used in the main text. Notes are descriptive, not interpretive.

$R(\gamma) = I$

Exact return. No remainder. No accumulation.

$e^{(i \cdot 2\pi)} = 1$

Non-exact return with cancellation. No accumulation.

$ds^2 = 0$

Exact return along null traversal. No accumulation.

$F = dA + A \wedge A$

Non-exact return with order dependence. Remainder accumulates.

Parallel transport around a loop

Comparison after traversal yields identity or holonomy. Accumulation encodes curvature.

$G\mu\nu = \kappa T\mu\nu$

Accumulated remainder fixed as constraint on traversal.